Agile Software Development

GERRY COLEMAN

Agile development provides an alternative to traditional sequential software development through employing customer collaboration and a whole-team approach, in which everyone has responsibility for quality, not just testers or others traditionally designed as quality-focused resources. Quality improvement and client satisfaction are sought by getting early and continuous feedback from the customer. The demands on a tester working on projects using agile methodologies are different to those faced on a traditional software project. In addition to excellent testing skills, testers must be able to communicate effectively, interact and collaborate constantly with colleagues, and respond to early and frequent customer feedback. This article describes the principles and values underpinning the agile development approach, which must be understood by all those wishing to be effective contributors on such projects.

This article is an edited excerpt from *Agile Testing Foundations: An ISTQB Foundation Level Agile Tester Guide* edited by Rex Black (ISBN: 9781780173368) to be published by BCS Learning & Development Limited in May 2017. It references the International Software Testing Qualifications Board syllabus for certifying Agile Testers (http://www.istqb.org/downloads/send/5-agile-tester-extension-documents/41-agile-tester-extension-syllabus.html).

KEY WORDS

agile principles, customer collaboration, feedback, testing, whole-team approach

INTRODUCTION

As the demand for new technological products grows apace, many companies have adopted agile methodologies for their software development activity. The demands on a tester working on projects using agile methodologies are different than those faced on a traditional software project. Testers on agile projects, in addition to excellent testing skills, must be able to communicate effectively, interact, and collaborate constantly with colleagues, liaise closely with business representatives, and work as part of a whole team responding to early and frequent customer feedback.

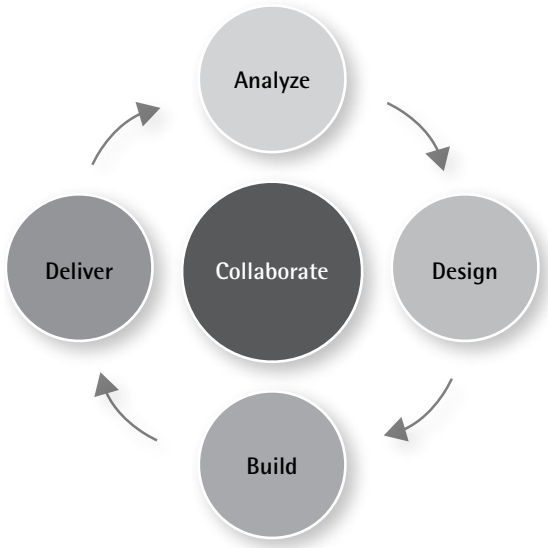
To work as part of an agile software team, testers must first understand the principles and values underpinning the agile development approach. This article looks at the essential components of agile software development, explains the advantages of the whole-team approach used in agile projects, and demonstrates the benefits of receiving early and frequent customer feedback.

THE FUNDAMENTALS OF AGILE SOFTWARE DEVELOPMENT

To function effectively as a tester on an agile project, one first needs to understand how agile software development works. Agile development, which covers a range of development models, is based on an "iterative" approach to creating software. In this way, the software product is built in progressive small chunks, with each new piece adding to the features developed in the preceding iteration.

Agile development focuses on early delivery of working software and aims to get the product into the hands of the customer as quickly as possible. Not only does the customer now receive a working product sooner in the development cycle, they can also provide early feedback on features, elements in the product they like or dislike, and aspects of the solution they wish to remove or modify. In this way, agile development can improve customer satisfaction and produce solutions that more closely meet customer needs. A simple agile development process is shown in Figure 1 on the next page.

FIGURE 1 Overview of the agile development process



AGILE SOFTWARE DEVELOPMENT AND THE AGILE MANIFESTO

As agile development emerged from a number of different models, the "Agile Alliance," a loose conglomerate of the model creators and protagonists, convened with a view to unify the various approaches around a common philosophy of shared principles and values. The outcome of these discussions was the "Agile Manifesto." The "Agile Manifesto" contains four values' statements:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

Agile proponents emphasize adhering to the values on the left, as they believe they have greater value than those on the right.

Individuals and Interactions

Looking at the first agile value statement, agile projects encourage communication between team members and between the development team and the customer. In traditional software development projects, the emphasis is often on ensuring the right processes are followed, and deviations from the process are detrimental and should not be encouraged. In addition, it is often argued that tools are seen as a solution to the logjams that can occur in traditional software development life cycles.

In agile, the focus is on the team members and their abilities. Confidence in the team to deliver is enhanced by the team's ability to interact. Testers in an agile project are an integral part of the development team, and it is through working together as a single unified team that product delivery targets are met. Don't assume that large, expensive tools are a key to success. With the right team, even the most "lo-fi" tools, such as whiteboards and sticky notes, can be significantly more effective than mandating an underperforming team use a computer-aided software engineering tool.

Working Software

One of the criticisms of traditional approaches to developing software is the strong emphasis placed on detailed documentation covering the outputs of each development phase. Agile development, by contrast, specifies that "just enough" documentation should be produced and that the focus instead is on creating working product early and often. The most important deliverable of any software project is the product itself. Agile prides itself on regular delivery of the desired features to the customer.

Rather than the agile team spending time producing detailed requirements specifications, design documents, test plans, and so on, the focus is primarily on creating the solution the customer desires. In teams that create lots of documents, a huge challenge is to keep the documents up to date, particularly where customer requirements are constantly changing or there is uncertainty in what the final product will look like. Agile teams, by contrast, create "just enough" documentation.

By transferring the emphasis from producing documentation to creating product, communication and interaction among all team members becomes essential. Without reports and templates being the repository of agreement, knowledge is shared verbally and the relationship with the customer deepened and enhanced.

Customer Collaboration

Too often in software development a document, such as a requirements specification, becomes the "de facto" contract between the software development team and the customer. As a contract, everyone is focused on getting this document watertight, removing any ambiguities, and achieving sign-off between both parties. This is typically done before the project starts.

In many cases, the next involvement the customer has is when the system is delivered and they are asked to conduct acceptance testing to confirm that what they specified in the contract is what has been delivered. When, as often happens, there are great divergences in what the customer believes they are getting and what they actually receive, the requirements specification is revisited and both parties now enter a confrontational phase over what the contract actually states.

Agile avoids this process by enshrining customer collaboration as one of its values. Through working closely with the customer and delivering the software on a regular basis, the agile team gets constant feedback, and any change requests can easily be incorporated before too much additional work is done. Treating the project sponsor as a partner in the project is a key tenet of agile development.

Responding to Change

Who has worked on a software project where customer requirements didn't change during the project? It would be incredible if, over even a three- to six-month development timeframe, customers didn't wish to modify existing requirements or incorporate new features into the product. Being able to respond to change in this way is essential for happy customers. Too many projects spend large amounts of time at the outset on creating detailed plans. As these projects progress, things happen that affect the project, for example, the customer's business environment changes, a competitor launches a similar product, or new legislation that will affect the product substantially is enacted. Any number of unforeseeable events can occur, which means the project must be easily able to accommodate changes. Agile, through the use of iterative development, focuses on general long-term release planning and short-term detailed planning. It's impossible to say what may happen during the product life cycle, so being able to respond to change quickly and effectively can be the difference between project success and failure.

THE AGILE PRINCIPLES

The four agile values drove the creation of 12 principles (see Figure 2), which sought to differentiate agile projects from traditional projects.

1. *Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.* Agile teams endeavor to deliver software on a frequent basis. Each version of the software includes additional features from the preceding version.
2. *Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.* In agile projects, change is considered inevitable. Change is seen as positive in that the customer, through seeing features early, has a better understanding of what they want.

FIGURE 2 12 principles of the Agile Manifesto

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, at intervals of between a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity—the art of maximizing the amount of work not done—is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

http://agilemanifesto.org/principles.html

3. *Deliver working software frequently, at intervals of between a couple of weeks to a couple of months, with a preference to the shorter timescale.* Giving the customer even a rudimentary solution, early in the project, starts the feedback process. This is then supplemented with a continuous supply of updated feature requests, thus deepening and enriching the customer's feedback.
4. *Business people and developers must work together daily throughout the project.* Agile projects depend on the continued involvement of the customer. Having a customer representative on site, during development, is the ultimate goal.
5. *Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.* People are the key to the success of agile projects, so remove any impediments to productivity and ensure the team is empowered to make decisions.
6. *The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.* The primary method of exchanging information within an agile team is through verbal communication, not documentation. It is impossible to capture every nuance of a feature in a document. Face-to-face communication clarifies requirements and helps eliminate ambiguity. Where the nature of the project makes face-to-face communication difficult, such as in distributed development, agile projects will then utilize web/videoconferencing, instant messaging, and telephone calls as substitutes.
7. *Working software is the primary measure of progress.* Rather than use phase completion as a yardstick for how much headway a project has made, agile development measures progress by how much functionality is working. This introduces the issue of "definition of done" whereby features are only said to be complete when they have passed customer acceptance tests.
8. *Agile processes promote sustainable development.* The sponsors, developers, and users should be able to maintain a constant pace indefinitely. Excessive overtime and continual pressure can result in mistakes, omissions, and, ultimately, employee burnout and disaffection. Agile teams aim for a sustainable pace that minimizes overtime and excessive hours. This approach ensures motivated team members and high-quality work.
9. *Continuous attention to technical excellence and good design enhances agility.* To achieve a high-quality end product requires a focus on quality all the way through development. Agile teams use good design techniques, and issues like redundant code are dealt with as they arise.
10. *Simplicity—the art of maximizing the amount of work not done—is essential.* Developing only the solution required to achieve the project goals is a key factor in agile projects. Agile teams do not try to incorporate every potential future requirement or possible request in the initial design. Instead, they focus on making the design as simple as possible. Thus, if the customer wishes to add a new feature, this design simplicity means the request can easily be accommodated.
11. *The best architectures, requirements, and designs emerge from self-organizing teams.* An agile team is an empowered team. The team self-organizes and decides how best to achieve project objectives. All team members share responsibility for the success of the project.
12. *At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.* The best software teams frequently engage in self-reflection. Regularly examining what's working and what's not working allows agile teams to remain agile.

WHOLE-TEAM APPROACH

Agile software development is conducted by cross-functional teams of people who possess a variety of skill sets. The teams typically house a range of expertise

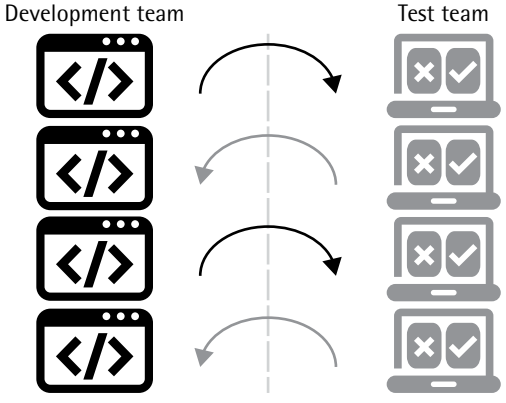
including programming, testing, analysis, database administration, user experience, and infrastructure, among others. All of this expertise is needed to deliver the product and is executed in agile projects by virtue of a whole-team approach.

Advantages of Using the Whole-Team Approach

Advantage 1: Making quality everyone's responsibility.

In traditional sequential software development, team roles are very clearly delineated. Quite often boundaries form between the various roles. This results in regular product handover from one group to another, for example, from developers to testers. With this product handover can also come responsibility handover. So, in this instance, when code is given to the tester by the programmer, it now becomes the tester's responsibility rather than the programmer's. When the tester finds failures in the delivered software, the product is handed back to the developer to be repaired. Thus, the code is now again the responsibility of the developer (see Figure 3).

FIGURE 3 Transfer of responsibility from developer to tester and vice-versa



All this back-and-forth activity can easily create division within software development teams. Animosities can arise, and testers are often seen as the "quality police," who are the conscience of the cavalier developers.

Agile development approaches software production differently. Within agile, timely product delivery is the responsibility of the whole development team. The agile team comprises all the skills needed by traditional software development teams. However, by creating one team with shared responsibility for quality and delivery, the walls that can divide the various roles are broken down, and collaboration becomes a necessity. Agile teams are therefore said to be cross-functional in that many different roles and specializations come together in a single team. Combining all the necessary skills together in a cross-functional team means the project benefits from the harnessing of this expertise. In agile, the whole (team) can truly be said to be greater than the sum of the parts.

Advantage 2: Enhancing communication and collaboration within the team.

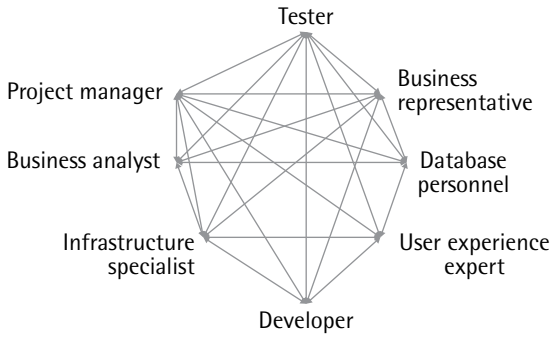
Where a cross-functional team is employed on an agile project, collaboration is key to success and is a fundamental component of the whole-team approach. To support close collaboration, agile teams are colocated. When everyone shares the same workspace, informal contact becomes straightforward. This is enhanced where the customer, or customer representative, is also on site.

Formal meetings and documentation are minimized in an agile project, so testers, developers, and other team members communicate constantly. One of the few formalities in an agile project is the stand-up meeting. Taking place daily, the stand-up meeting involves the whole team and is used for status reporting of work. By using this simple approach, each team member is kept up to date on what everyone else is doing. However, the stand-up meeting does not preclude the need for individual face-to-face communication by team members. Face-to-face communication helps remove the impediments highlighted in the stand-up meeting and ensures project momentum is maintained.

Testers will still need to speak to business representatives, however. Business analysts may need to speak to user experience experts. Network specialists may need to speak to developers. The potential communication channels are many and varied (see Figure 4 on the next page). All this communication not only helps to share understanding about the product but also helps share responsibility for it too. Agile projects benefit greatly from this communication and collaboration.

Agile projects deliberately encourage collaboration through the use of user stories, which are very short descriptions of system features. User stories provide just enough information to allow planning and estimation of

FIGURE 4 Potential communication channels in an agile team



work to take place but not enough for the feature to be developed and tested. For this to happen, it is necessary to speak to the person, from the business side, who is proposing the feature. Only they, as the customer, or customer representative, know exactly how the feature should work, so collaborating closely with them greatly aids in understanding the requirement.

Advantage 3: Enabling the various skill set within the team to be leveraged to the benefit of the project.

Testers on an agile team work with the business representatives to develop suitable user acceptance tests. Only the business representative will know exactly how the system will function. Working together with the business sponsor, the tester can assist in creating acceptance tests that help the team determine when a system feature is complete or done. In agile projects many of these acceptance tests are automated, so there is great onus on the tester to ensure feature requests from business representatives are as testable as possible and can be easily encoded into a tool.

As part of their role within the whole-team approach, testers will also work closely with developers. Testers can help developers create automated unit tests but have an even greater role to play where continuous integration (CI) is used. Daily, or more frequent, builds of software demand continuous regression testing. For CI to be effective, the regression tests must be automated. Working together, developers and testers can utilize their skills to create an automated regression test suite that is used each time a new system build is created. In addition, the test suite must be easily modified and added to, as new functionality is integrated and thus new corresponding tests are required.

Through collaboration with business representatives and developers, testers can help share knowledge about testing. Cross-team working increases knowledge sharing, and all team members get a better understanding of each other's work.

One of the drawbacks with traditional software development projects is that developers and testers frequently have no interaction whatsoever with the customer. Much of the product information they receive is through project documents or from conversations with other development team members such as business analysts. Encouraging testers and developers to collaborate directly with customers or customer representatives means system features are much better understood than would otherwise be the case.

Crispin and Gregory (2009) propose using the "Power of Three" for such team interactions. Any feature meeting or discussion should involve a business representative, a developer, and a tester. The presence of all three roles in discussions provides a shared clarity and understanding of system features within the team.

EARLY AND FREQUENT FEEDBACK

In sequential development projects, the development cycle can be long, and customer involvement in the process may be limited, with most feedback coming late in the process. Agile development benefits from early and frequent feedback from the customer.

Benefit 1: Avoiding requirements misunderstandings that may not have been detected until later in the development cycle when they are more expensive to fix.

In software projects that use traditional methods, customers are heavily involved in the early phases of the process, during requirements capture and definition. Once the customer has signed off on the requirements document, the next major input they may have is at the later stages of product testing when it is too late to remedy requirements misunderstandings or misinterpretation. It is also very expensive to make changes at this stage, as the amount of rework may be substantial. Continual customer involvement resulting in early and frequent feedback helps eliminate requirements ambiguity.

Benefit 2:
Clarifying customer feature requests, making them available for customer use early. This way, the product better reflects what the customer wants.

Agile projects use short iterations during the development cycle. Within these short cycles, feature clarification, design, code, and test take place. At the end of the cycle, the team has produced a "minimum viable product." Each subsequent development iteration adds to the product until eventually the entire system is complete. Business representatives on the project assist in feature prioritization resulting in the most important features, those with the highest business value, being developed first. This also reduces system risk, as one always knows he or she is working on the system aspects that have greatest customer worth.

Benefit 3:
Discovering (via continuous integration), isolating, and resolving quality problems early.

Projects that use sequential software development approaches perform integration testing toward the end of the development life cycle. Any integration problems, or system interface issues, only become apparent late in the process and can be a major headache to fix. Using continuous integration, agile projects benefit from the fact that any integration issues are instantly highlighted, the module(s) concerned can be isolated pending repair, and the system rolled back to the previous working version. When the fix is made, the module can then be reintegrated into the system and retested. Quality problems are resolved early before there is additional downstream system damage.

Benefit 4:
Providing information to the agile team regarding its productivity and ability to deliver.

Early and frequent feedback from the customer provides knowledge to the agile team on its productivity. Each feature is estimated and prioritized prior to development and then built during a development iteration. When the feature passes the acceptance tests it can be classed as "done." This process allows the agile team to measure how much product it can deliver per iteration. Measures such as these demonstrate the team's productivity, or velocity, and greatly assist with subsequent estimation and release planning.

In addition, the agile team can see what is hindering their productivity and progress allowing them to take remedial action.

Benefit 5:
Promoting consistent project momentum.

Project team morale is improved by the visibility of progress. Getting early and frequent feedback from the customer makes all parties more engaged. Satisfying the customer is very rewarding for the agile team.

SUMMARY

Agile development provides an alternative to traditional sequential software development. Agile promotes the benefits of working software, customer collaboration, the ability to respond to change, and values talented individuals and encourages them to interact continually to the benefit of the project. The agile philosophy is supported through 12 key principles.

By making the customer requirements central, agile seeks to ensure complete customer satisfaction with the finished system. Agile uses a whole-team approach, whereby everyone has responsibility for quality, not just testers. Quality is improved, and client satisfaction is achieved, by getting early and continuous feedback from the customer.

REFERENCE

Crispin, L., and J. Gregory. 2009. *Agile testing: A practical guide for testers and agile teams.* Reading, MA: Addison Wesley.

BIOGRAPHY

Gerry Coleman is a member of the Computing and Mathematics Department at Dundalk Institute of Technology, Ireland, where he specializes in teaching software engineering, with a particular focus on agile methodologies and software testing. He has more than 30 years of experience in the IT industry as a practitioner, academic, and researcher, and he received his doctorate from Dublin City University for research into software process improvement in the indigenous Irish software industry. Prior to taking up an academic position, he worked for a number of years as a software developer, systems/business analyst, and project manager in a number of financial institutions and was a senior consultant, with responsibility for research, and technology training and transfer, in the Centre for Software Engineering at Dublin City University. Coleman can be reached by email at gerry.coleman@dkit.ie.